QA Consulting

JavaScript Exercise Book

TRAINING MATERIALS

Contacts

elliot.womack@qa.com

team.qac.all.trainers@qa.com

www.consulting.qa.com

CONTENTS

Level 1 - Basic

CREATING A SCRIPT

Creating a HTML File that has the following JavaScript code nested within a **<script>** tag

```
alert("Hello World!");
```

Open the HTML file in a browser, it should present a modal dialog with **"Hello World"** inside of it.

VARIABLES

Create a variable using the **var** keyword, store a value in it.
Output the contents of that variable using three different methods

```
alert()
console.log()
document.write()
```

FUNCTIONS 1

Create a function that accepts a number and **returns** its square

Input-> 2 Output -> 4

Call your function and output your result via one of the three methods.

FUNCTIONS 2

Create a function that accepts 3 numbers and returns the sum of them

Input-> 1,2,3 Output->6

VARIABLES 2

Create a Person object with three attributes. **name, age,** and **occupation.**

Output the contents of your object, then edit your object, then output them again to show the changes.

EVENTS 1

Create a HTML button that calls a function when clicked.

The function that it calls should increase the age of your person object.

FUNCTIONS 3 / EVENTS 2

Create a series of textboxes and buttons that let you create a Person object with data from the textboxes, output your Person object, as well as edit your Person object.

STRINGS 1

Create a variable with `" He said "My name is Elliott" "` as the value, and display it.

Using **String Methods**, convert this string to uppercase, and display it.

STRINGS 2

Concatenate a String and a Number together, and output the result.

ARRAYS 1 / STRINGS 3

Create an array with 3 strings inside of it, and output them.

Then add another string to your array, and output them.

Then remove the last string from the array, and output them.

CONDITIONALS 1

Using conditional statements, create a function that checks if your person object is between 20-40 years old, then outputs if that fact is true or not.

ITERATION 1

Create a **for** loop that increments from 1 to 10, outputting the current iteration at each step.

ITERATION 2

Create a conditional statement inside your for loop body, which then checks what the current iteration is, if the current iteration is divisible by 2, then it outputs it, otherwise it outputs nothing.

Level 2 - Intermediate

ITERATION 3 – FIZZ BUZZ

Create a method that prints the numbers 1 to 100 via a for loop, however if the number it's outputting is divisible by 3 then output "Fizz" instead, if the number is divisible by 5 output "Buzz" instead. If the number is divisible by both 3 and 5, output "FizzBuzz"

Additional – Add 3 additional parameters to your fizz buzz method, one for the number it will count up to, one for the word it outputs if its divisible by 3 and one for the word it outputs is divisible by 5, then make your application work with these parameters instead of hard-coded values.

ITERATION 4

Create a method accepts a number then, outputting what it did at each iteration, and how many iterations it took to get to the end.

- If the number is divisible by 3, divide it by 3.
- If it's not, either add 1 or subtract 1 (to make it divisible by 3), then divide it by 3.
- It stops when the number it reaches is 1

STRINGS 4

We'll say that a "triple" in a string is a character appearing three times in a row. Create a function that accepts a string and returns the amount of times a triple exists in that string.

Triples may overlap.

Input-> abcXXXabc Output-> 1

Input-> xxxabyyyycd Output-> 3

DOM 1

Create a function that creates a paragraph tag

Create another function that changes the text of that paragraph tag to what is in a textbox

Create another function that deletes the paragraph tag

All of these functions should be called with a button(s)

JSON 1

Using basic JavaScript, access the JSON file at
https://raw.githubusercontent.com/ewomackQA/JSONDataRepo/master/example.json

Once loaded, display the objects retrieved in an organised fashion.

JSON 2

Using basic JavaScript, access the JSON file at
https://raw.githubusercontent.com/ewomackQA/JSONDataRepo/master/kings.json

Once loaded, create a search feature that will search for a specific king(s) depending on the input

Input > **House of Denmark**

Output > **Cnut**
United Kingdom
House of Denmark
1016-1035
Harold I Harefoot
....

GARAGE

Create a virtual garage with the following functionality;

- Check in cars to the garage
- Check out cars from the garage
- Output the contents of the garage
- Calculate the bill for a car, dependant on its attributes

All of this functionality should be done via a user interface, buttons and textboxes and any other form elements you think are appropriate.
Extra functionality that should be created, not garage related;

- Creating cars with no faults
- Creating cars with numerous faults

These will also be done via a user interface, however it should be distinguished to be separate from the Garage.

ADMIN INTERFACE

Following on from the initial garage application, create a console interface that accepts commands from a textbox, simulating a CLI.

e.g.
create car Peugeot yk10ykm 4 NOFAULTS
output Garage
check in yk10ykm
check out yk10ykm

All the functionality that you previously provided in your interface should be possible to be used through this interface

The commands must be one-liner statements, no menu systems of the sorts.

Level 3 - Advanced

REKTANGLES

DESCRIPTION

Given a word, a width and a length, you must print a rectangle with the word of the given dimensions.

FORMAL INPUTS & OUTPUTS

INPUT DESCRIPTION

The input is a string word, a width and a height

OUTPUT DESCRIPTION

Quality rectangles. See examples. Any orientation of the rectangle is acceptable

EXAMPLES

INPUT

```
"REKT", width=1, height=1
```

OUTPUT

```
R E K T
E     K
K     E
T K E R
```

INPUT

```
"REKT", width=2, height=2
```

OUTPUT

```
T K E R E K T
K     E     K
E     K     E
R E K T K E R
E     K     E
K     E     K
T K E R E K T
```

Bonus challenge – Make it output in different shapes. E.g. Hollow Diamond

CODE CLEANING

Four exercises, to clean and fix some code, conforming to certain standards detailed in each readme file of the exercise.

https://github.com/ewomackQA/JavaScript-Exercises

For these exercise you must use GIT, commit regularly and feed your git repository through www.codeacy.com . Once each exercise is completed upload a screenshot of the codeacy result to the project folder.